C and Arduino

OUR MAIN TOPICS TODAY

Preprocessor, Include

I/O Operations

Data types

Operators

Control flow

Arrays

Functions

Breadboard

Resistor

LED

Potentiometer

Buzzer

PREPROCESSOR, INCLUDE

- Analyse preprocessor statements before compiling source code
- Presence of "#" sign as first non space character in line
- Can define constants, macros, write conditionals and include header files
- Examples: #ifndef, #ifdef, #include, #define

I/O OPERATIONS

printf("Welcome to ESCENDO 2022");

int i; scanf("%d", &i);

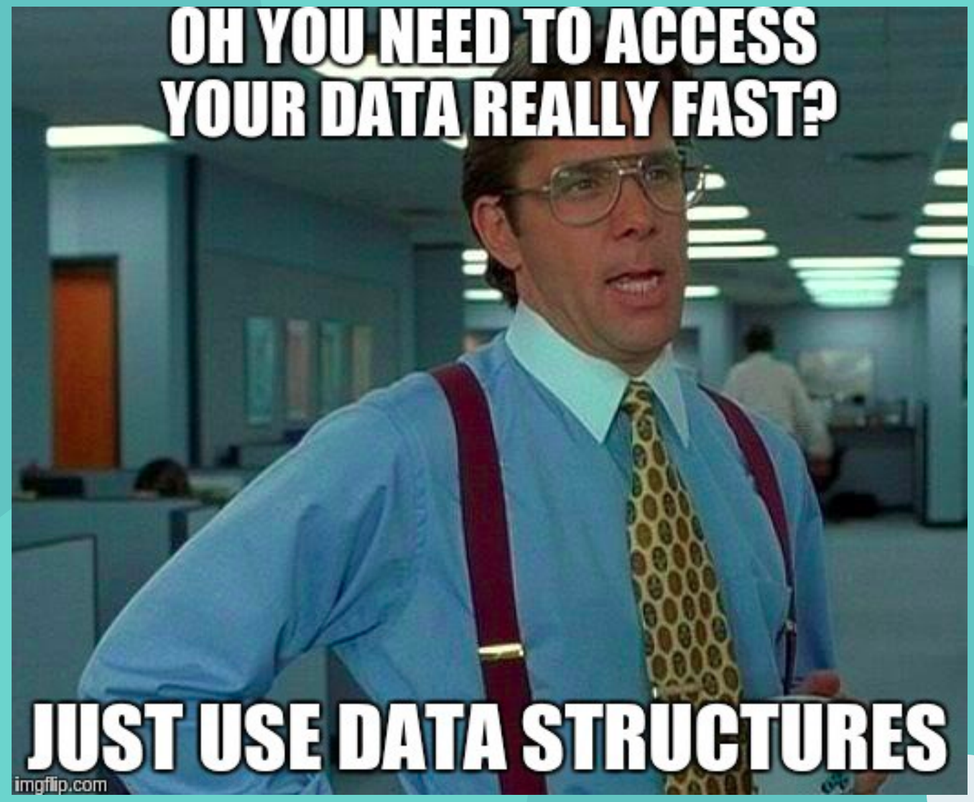
- printf(): Print outputs to standard output stream (Useful for primitive debugging)
- scanf(): Scan input from standard input stream according to format specified

Data Types

- Variables are names given to computer memory locations, to store values
- A data type represents a type of data that can be processed, also determines the byte size
- Declare type of variable and initialize
- Examples: int, float, double, char, _Bool

type-specifier variable-name;

```
int a;
int a, b, c;
int a=5;
int a=4, b=6;
```



DATA TYPES

- int: integer, can be +ve, -ve, 0 (int a=0x15; //hexademical)
- float: floating point numbers (float a=-0.123;)
- double: floating point numbers with 64 bits precision
- _Bool: true or false (_Bool a=1;)
- unsigned, long, short

- enum: specify a variable and valid values that can be stores

```
enum bodyParts {eyes, teeth, nose};
enum bodyParts part1;
part1=eyes;
```

- char: letter of digit (char alphabet; alphabet='A';)

- format specifiers: used to display variables as o/p (int a=4; printf("a is %d", a);)
- int: %d, float: %f, char: %c, _Bool: %i, double: %e, %g

OPERATORS

- Mathematical and logical operations
- Example: assignment, relational (>,<,!=), bitwise(>>,<<)
- Arithmetic: +, -, *, /, %, ++, --
- Logical: &&, ||, !
- Assignment: =, +=, -=, *=, /*, %=, <<=, >>=
- Relational: ==, !=, >, <, >=, <=
- Bitwise: &, |, ^, ~, >>, <<

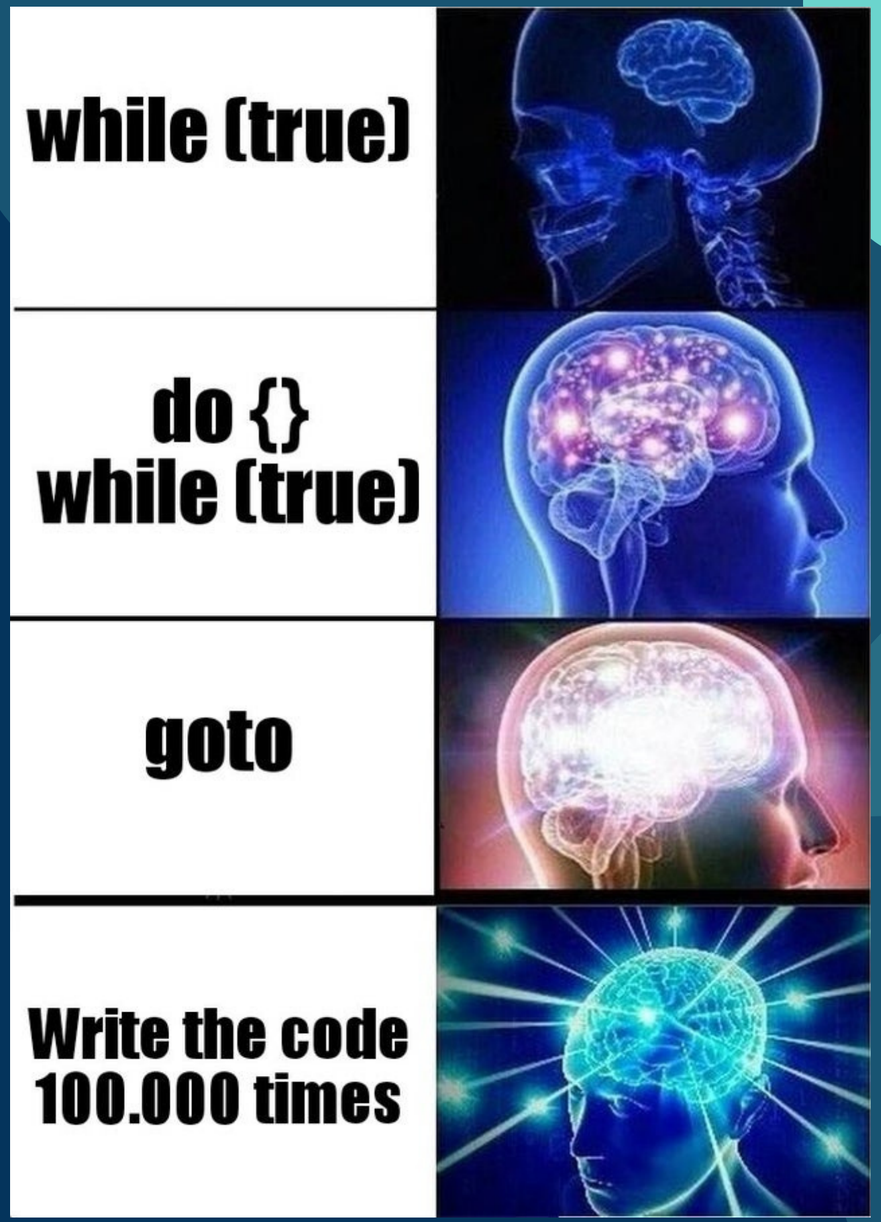
- cast: (int) 24.98 + (int) 67.3 Result: 24 + 67 = 91
- sizeof: Bytes used by a variable of specific type in memory
- ?: (Ternary opearator) If True? then x: otherwise y
- *: Pointer

OPERATORS PRECEDENCE

- Decides the order of operator execution by grouping terms in an expression and evaluating them
- The operator precedence in C can be found here: https://en.cppreference.com/w/c/language/operator_precedence

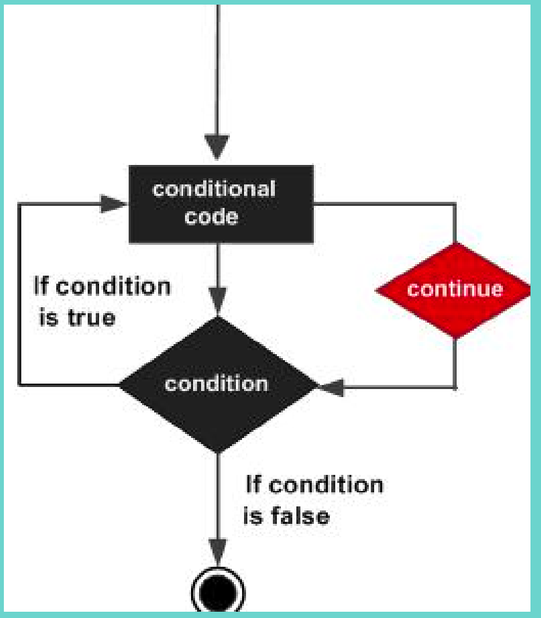
CONTROL FLOW

- Control flow statements break the normal top to bottom flow of execution
- Conditionally execute certain blocks of code
- Decision making statements (if, if-else, switch, goto)
- Looping statements (for, while, do-while)
- Branching statements (break, continue, return)



CONTROL FLOW

```
if ( expression 1 )
    program statement 1
else if ( expression 2 )
    program statement 2
else
    program statement 3
```

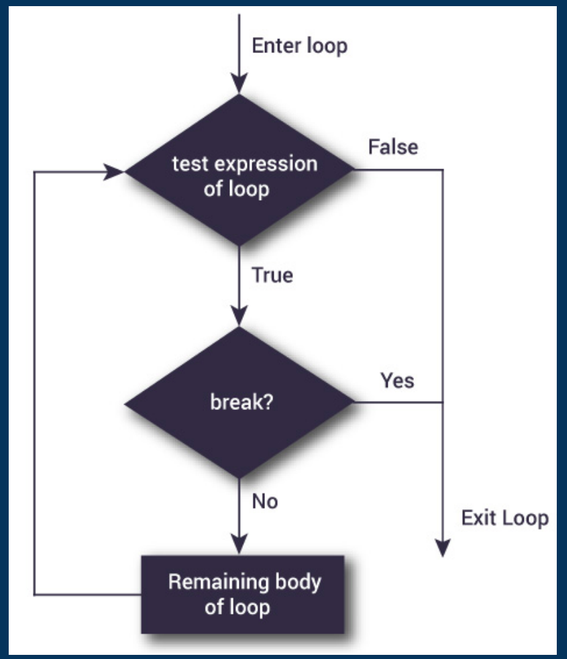



```
for(starting_condition; continuation_condition ; action_per_iteration)
  loop_statement;
```

```
switch ( expression )
{
    case value1:
            program statement
          ...
            break;
    case valuen:
            program statement
            program statement
                ...
            break;
    default:
            program statement
             ...
            break;
}
```

```
while (expression)
  {
            statement1;
            statement2;
    }
```

```
do
    statement
while ( expression );
```



ARRAYS

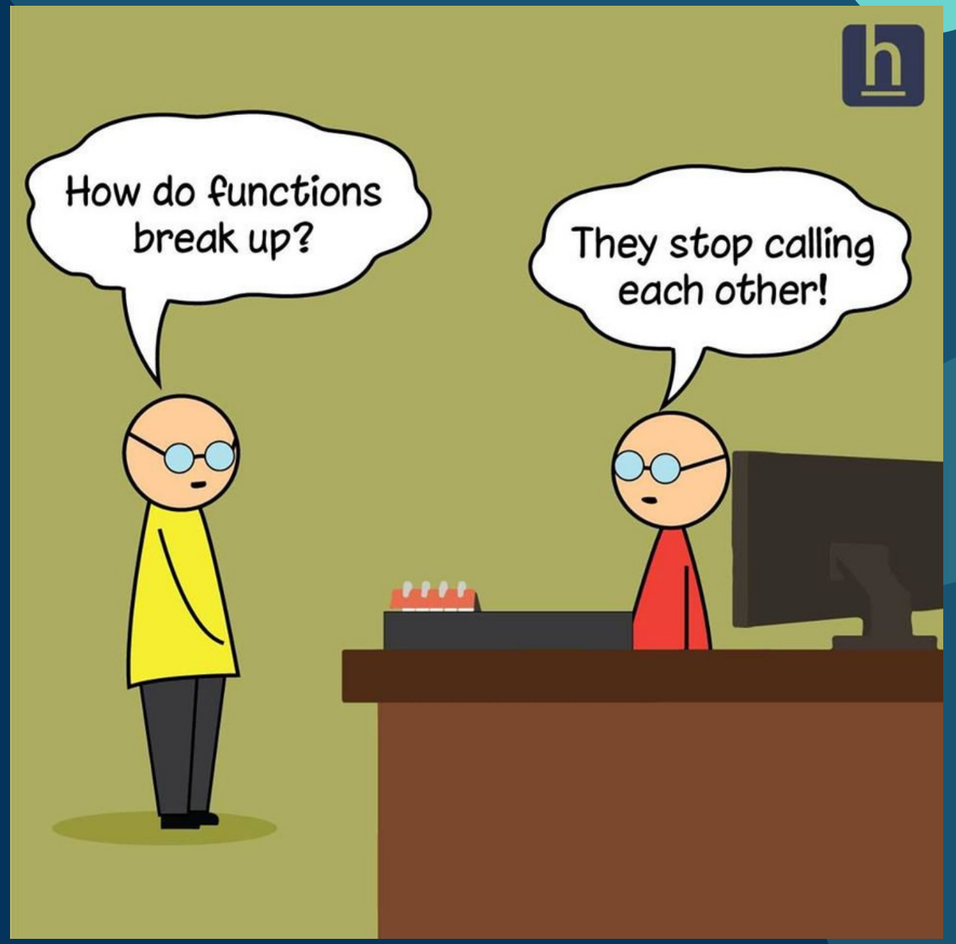
- Data structure to store many elements of the SAME data type
- int counter[5]={1,2,3,4,5}; float data[500]={100.2, 201.7};
- Create multidimensional arrays

int matrix[2][3]={{1,2,3}, {4,5,6}};

- The number of elements mentioned in the square brackets should be a positive integer>0, or any expression evaluating to the same

FUNCTIONS

- Self contained modules of code that accomplish a specific task
- Reduces overall complexity, duplication of code and improves readibilty of code
- main() functions is the start of execution of a program in C, allows to pass in command line arguments and output data
- Local variables are local to the function
- Global variables are global to the program and have the lifetime of a program
- Good coding practice is to generally avoid using global variables



FUNCTIONS

```
Return_type  Function_name( Parameters - separated by commas )
{
   // Statements...
}
```

Definition of a function

- Function prototypes is a statement that defines the name, return value type and the type of each of the parameters of the function
- A function that returns nothing has a return value type of void

```
//Function prototype
double add (double a , double b);

//Function definition
double add (double a , double b)
{
    return a+b;
}
```

```
int main()
{
   float a=3, b=5;
   float c;
   c=add(a,b);
   printf("The sum is %f\n", c);
   return 0;
}
```

ARGUMENTS AND PARAMETERS

- Parameters help pass data into the function
- The assume the value of the arguments passed when calling the function and remain local to the function
- Arguments are values that can be passed to the parameters of the function when the function is called

THANK YOU!

Feeling confused...?

Ask us!

CREDITS

1. https://www.freecodecamp.org/news/trees-in-programming-the-oxygen-of-efficient-code-6c7c11a3dd64/
2. https://morioh.com/p/1f1cc4ba5319
3. https://www.reddit.com/r/ProgrammerHumor/comments/7d3szb/infinite_loop/
4. https://www.facebook.com/103951884287338/posts/dont-shoot-im-a-programmerprogrammer-memes-funny/283287173020474/
5. https://www.pinterest.com/pin/143059725641565518/
6. https://me.me/t/parameters